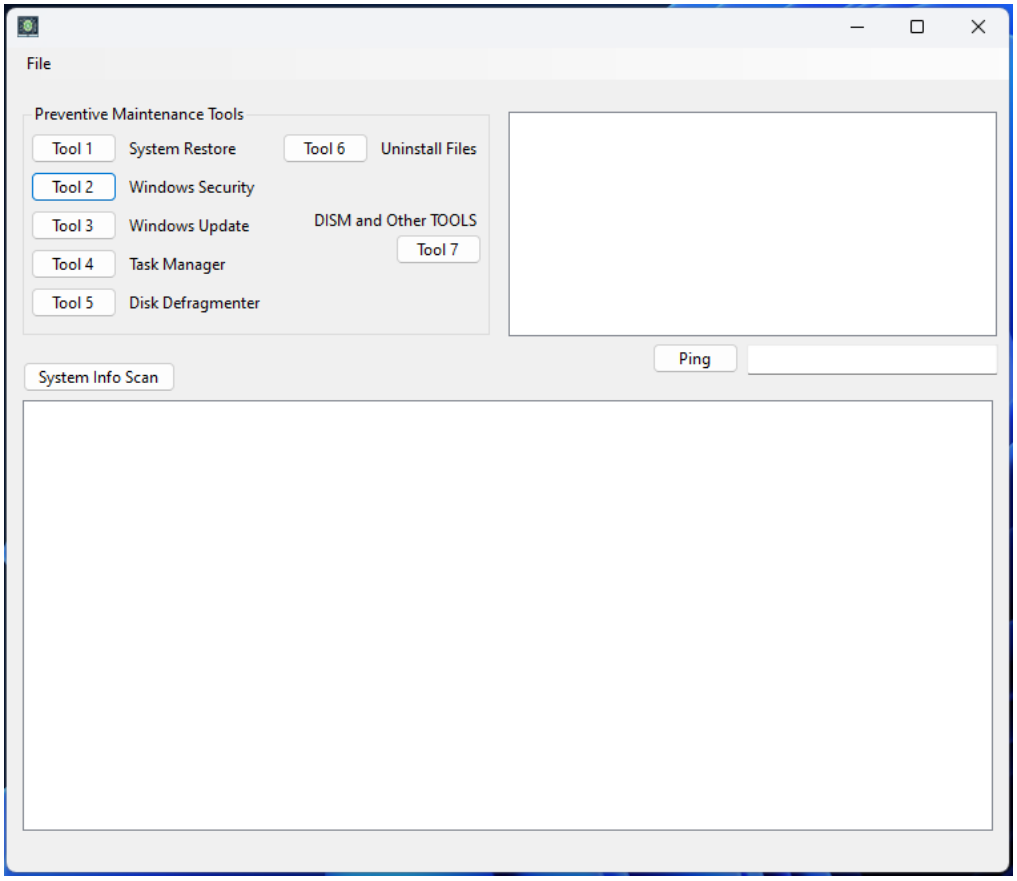
Nia Region 3 Preventive Maintenance Toolkit for Desktop and Laptop

The NIA Region 3 Preventive Maintenance Toolkit for Desktop and Laptop is a VB-developed application designed to support IT staff and designated personnel in performing annual preventive maintenance on ICT equipment. This toolkit comprises seven distinct tools, each aimed at streamlining and accelerating the preventive maintenance process for ICT devices.



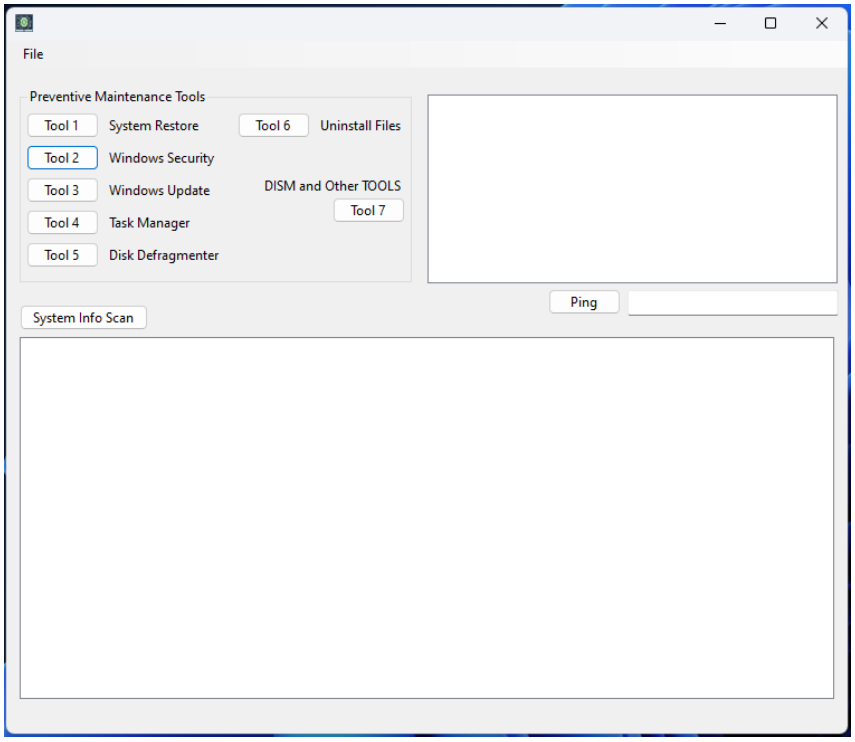
Installation guide:

The application operates as a portable solution, requiring only that the ICT equipment has Windows Runtime 6.0 installed. If the runtime is not present, the installer can be freely downloaded from Microsoft, ensuring easy setup and use. Additionally, the IT staff can store the installer on a USB storage device for easy installation. The application is compact, with a size of under 57 MB, making it convenient to transport and deploy across multiple machines.

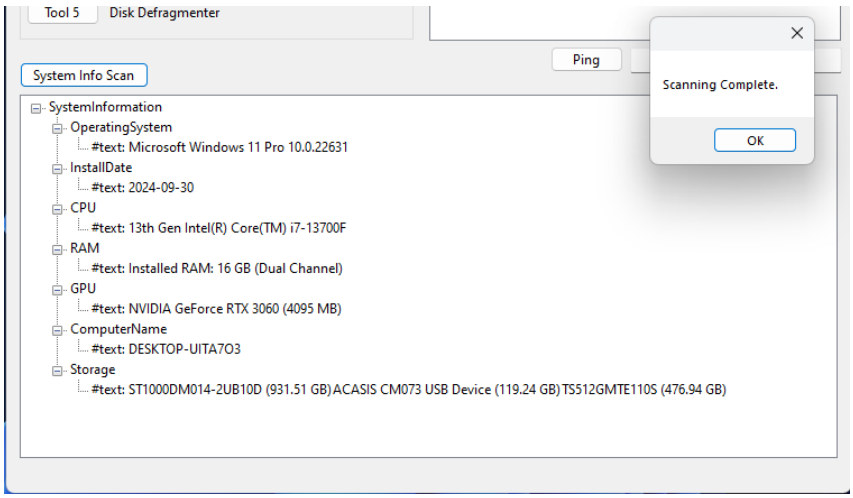
Main Interface

The main form of the toolkit provides easy access to all available tools through a series of buttons.

Each button triggers the corresponding tool as detailed in the description on the right side of the interface.



A small window within the main user form that, when activated, performs a comprehensive scan of the system unit to retrieve current settings and specifications. The scan provides detailed information, including:



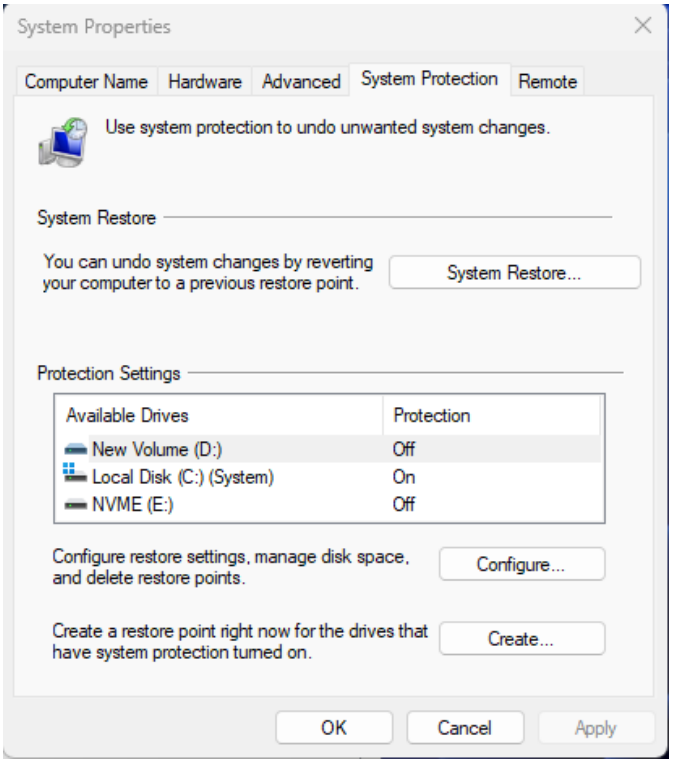
- Operating System (OS) Installed: Identifies the current OS and its version.
- Installation Date: Determines when the OS was installed.
- CPU Details: Information about the installed Central Processing Unit.
- RAM Information: Amount of RAM installed in the system.
- GPU Details: Information about the installed Dedicated Graphics (GPU).
- Computer Name: The current name assigned to the computer.
- Storage Devices: Lists all storage devices present during the scan, including internal drives and connected USB storage devices.

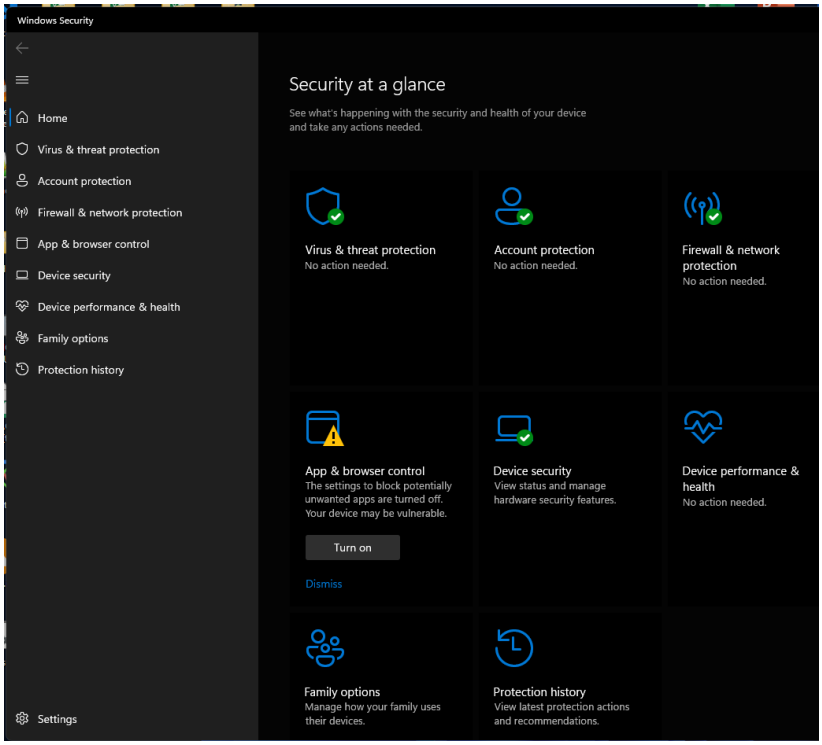
Tool 1 launches System Properties window,

This feature allows you to create a system restore point for the machine, named with the pattern "PM-YYYY" (where YYYY represents the current year). It ensures that the system protection settings are correctly configured, specifically:

- Activation: The system protection for the drive labeled as "System" must be turned on.
- Storage Allocation: At least 10% of the drive's space is allocated for system restore purposes.

By adhering to these settings, you can safeguard your system's stability and quickly revert to a previous state if needed.

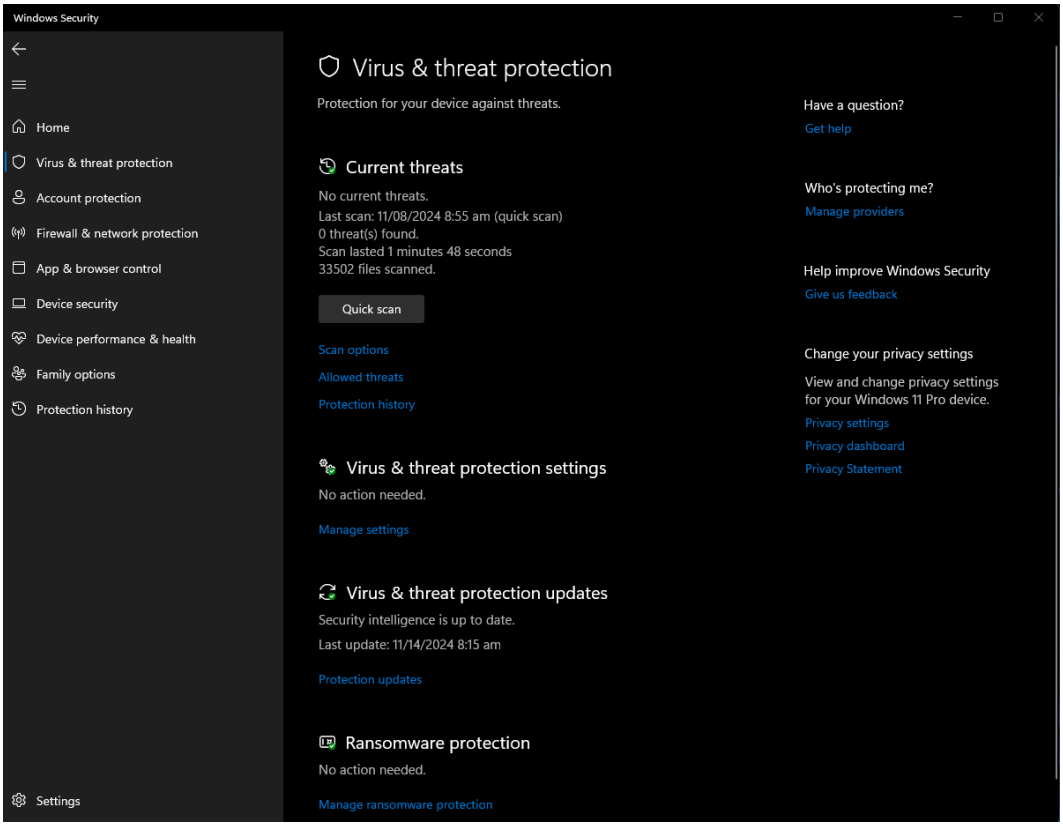




Tool 2 launches Windows Security Management,

This feature provides guidance on managing your system's security settings:

- For Paid Third-Party Security Programs: Ensure the security program is automatically updated to maintain optimal protection.
- For Free Third-Party Security Programs: Uninstall the free security program and switch to the built-in Windows Security. Make sure to enable all protection settings within Windows Security for comprehensive coverage.

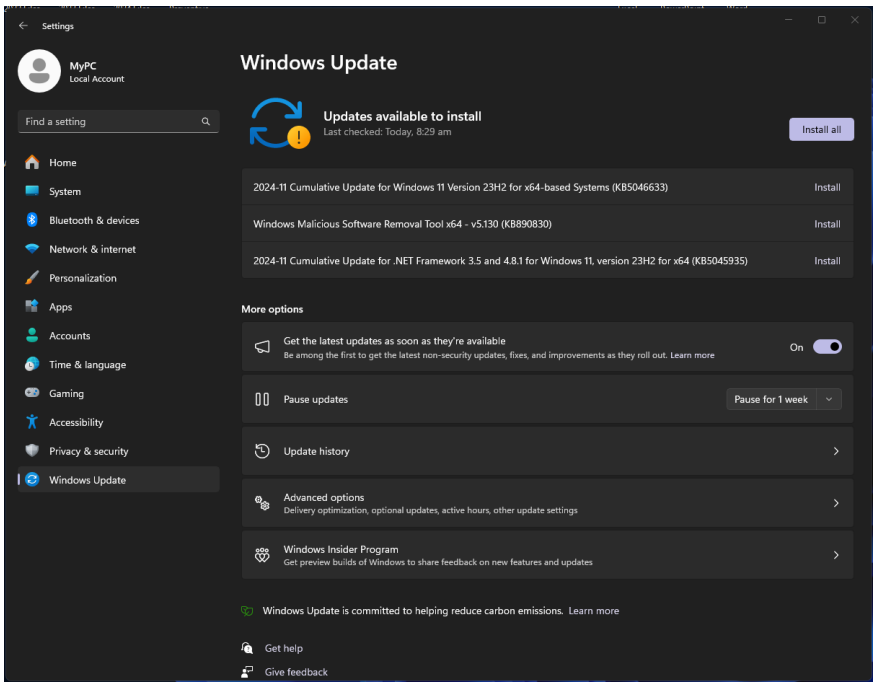


Virus and Threat Protection under windows security ensure your device's virus and threat protection is always up-to-date and regularly perform quick scans. Due to the potentially lengthy duration of scans based on the number of files, it is not recommended to wait for completion. Instead, advise end users to notify IT staff once the scan is finished.

Tool 3 launches Windows Update,

Ensure that Windows Update is enabled and encourage users to update their systems regularly, particularly focusing on the Windows Malicious Software Removal Tool. Advise users to avoid interrupting updates to prevent software malfunctions.

For users with outdated operating systems that no longer receive updates, recommend installing a third-party internet security program and consider upgrading the system if possible.

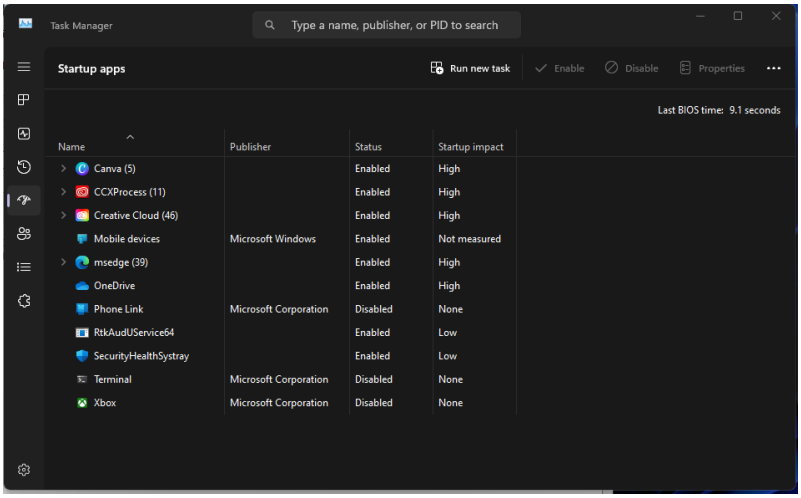


Tool 4 launches Task Manager,

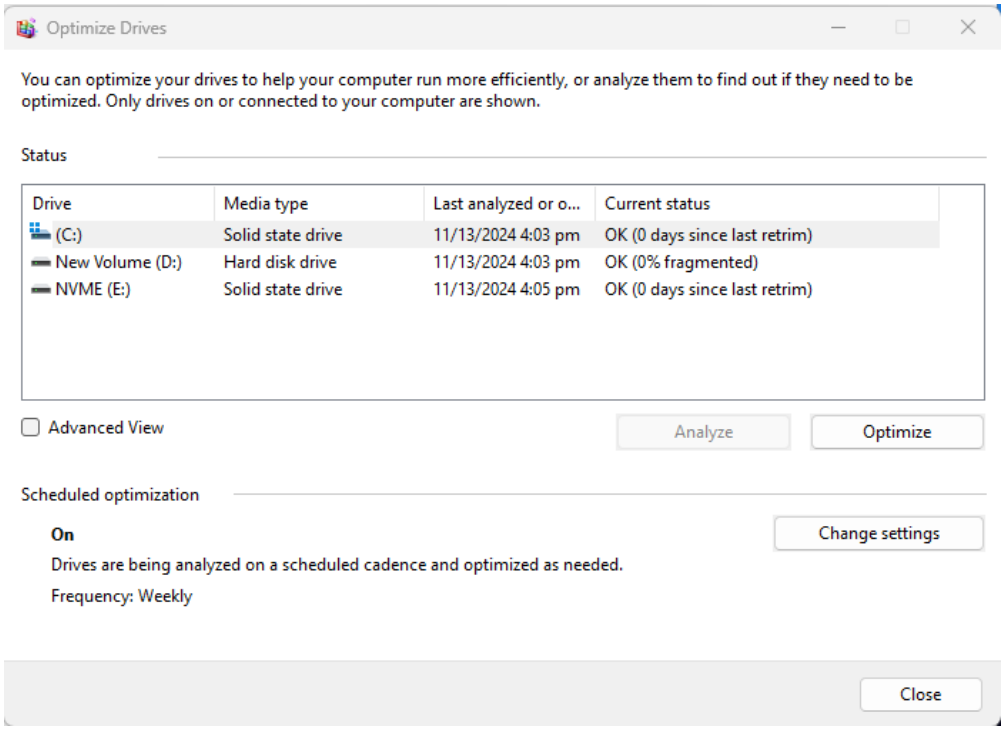
Task Manager - Startup Apps Management

Monitor startup applications for their impact on system performance. For each application:

- Essential Applications: Keep enabled if necessary for system functionality.
- Non-Essential Applications: Disable if not critical to system operation, unless actively used by the end-user.

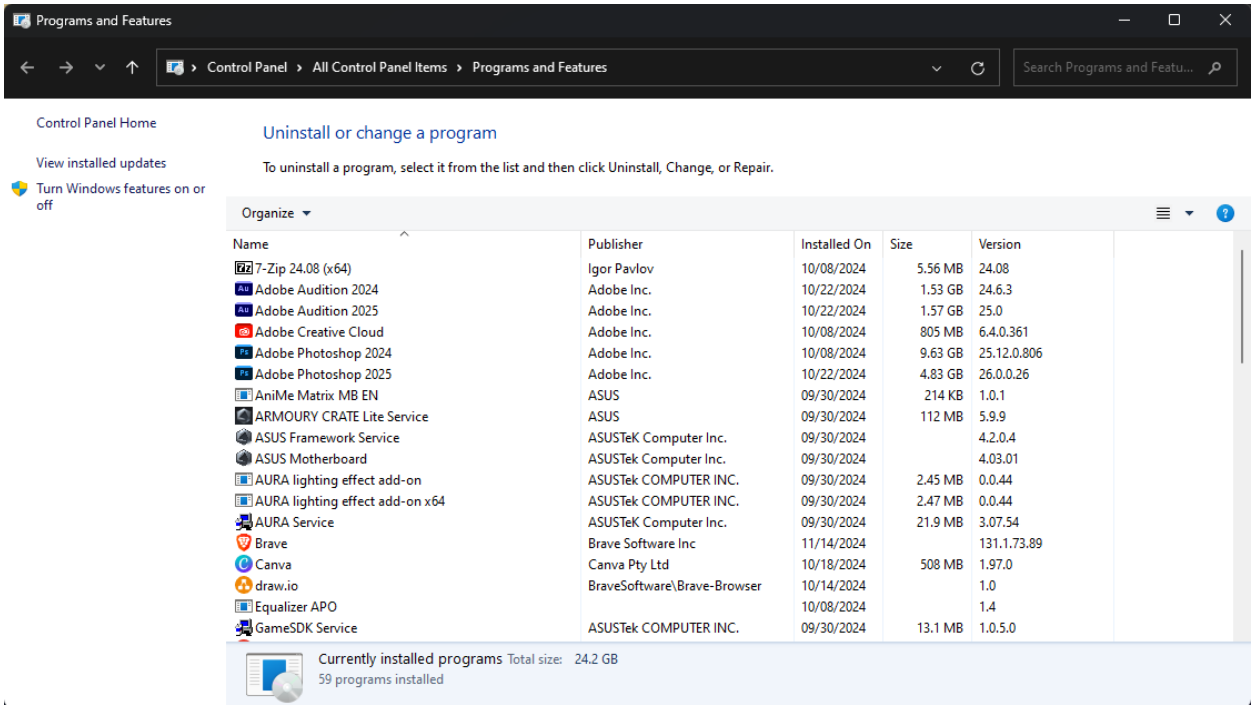


Ensure that all non-work-related applications (e.g., Spotify, Xbox, etc.) are disabled to optimize system performance and efficiency. This approach helps maintain a smooth and responsive system by minimizing unnecessary startup load.



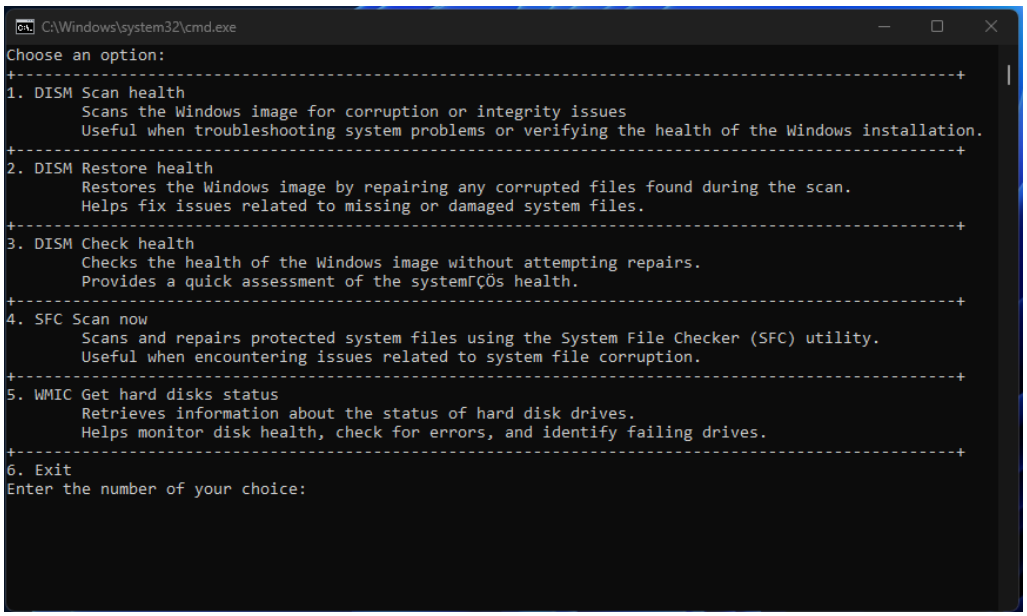
Tool 5 launches Disk Defragmentation.

Ensure that drive optimization is enabled and set the frequency to at least weekly. This regular maintenance helps improve system performance and longevity by keeping the storage drives in optimal condition.



Tool 6 launches uninstall and change a program,

Manage installed programs by uninstalling any unnecessary applications that do not contribute to work-related tasks, such as games and software with cracked licenses. If additional software is required, prefer open-source applications that meet the necessary needs. This ensures a streamlined and compliant software environment on the system.



Tool 7 launches an external Command Prompt window with full description of their functionalities.

System Requirements

RAM: At least 40 Mb available
CPU: At least 1 Ghz
Operating System: Minimum of Windows 8

This application is coded by NIA Region 3 Regional Office IT Staff, with few help from selected GPTs. No paid application or external application is used in the application all tools are available inside windows environment.

Prepared and Submitted by:

Achilles S. Bitangcol
Senior Computer Operator

Reviewed by:

Khristalyn M. Santiago
Senior Computer Services Programmer

Noted by:

Josephine B. Salazar
Regional Manager

Source Code:

```
Imports System.IO
Imports System.Management
Imports System.Xml

Public Class Form1
  Private Sub Form1_Load(ender As Object, e As EventArgs) Handles MyBase.Load

  End Sub
  Private Sub nfoScan_Click(sender As Object, e As EventArgs) Handles nfoScan.Click
    Try
      ' Collect system information
      Dim systemInfoXml As String = GenerateSystemInfoXml()

      ' Display XML in TreeView
      PopulateTreeViewWithXmlData(systemInfoXml)
      MessageBox.Show("Scanning Complete.")
    Catch ex As Exception
      MessageBox.Show("An error occurred: " & ex.Message)
    End Try
  End Sub

  Private Function GenerateSystemInfoXml() As String
    Dim xmlDoc As New XmlDocument()
    Dim xmlDeclaration As XmlDeclaration = xmlDoc.CreateXmlDeclaration("1.0", "UTF-8",
Nothing)
    xmlDoc.AppendChild(xmlDeclaration)

    Dim rootNode As XmlNode = xmlDoc.CreateElement("SystemInformation")
    xmlDoc.AppendChild(rootNode)

    ' Add collected information to XML
    AddXmlElement(xmlDoc, rootNode, "OperatingSystem", GetOperatingSystemInfo())
    AddXmlElement(xmlDoc, rootNode, "InstallDate", GetOperatingSystemInstallDate())
    AddXmlElement(xmlDoc, rootNode, "CPU", GetCpuInfo())
    AddXmlElement(xmlDoc, rootNode, "RAM", GetRamInfo()) ' Ensure this captures the
numeric value
    AddXmlElement(xmlDoc, rootNode, "GPU", GetGpuInfo())
    AddXmlElement(xmlDoc, rootNode, "ComputerName", Environment.MachineName)
    AddXmlElement(xmlDoc, rootNode, "StorageDevices", GetStorageInfo()) ' Changed to
StorageDevices

    Return xmlDoc.OuterXml
  End Function

  Private Sub PopulateTreeViewWithXmlData(xmlData As String)
    treeViewXML.Nodes.Clear()
    If Not String.IsNullOrEmpty(xmlData) Then
      Dim xmlDoc As New XmlDocument()
      xmlDoc.LoadXml(xmlData)
      Dim rootNode As TreeNode = treeViewXML.Nodes.Add(xmlDoc.DocumentElement.Name)
      AddNode(xmlDoc.DocumentElement, rootNode)
      treeViewXML.ExpandAll()
```

```
    Else
      MessageBox.Show("XML data is empty.", "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error)
    End If
  End Sub

  Private Sub AddNode(xmlNode As XmlNode, treeNode As TreeNode)
    For Each childNode As XmlNode In xmlNode.ChildNodes
      Dim childTreeNode As TreeNode

      ' Handle text nodes directly
      If childNode.NodeType = XmlNodeType.Text Then
        treeNode.Text &= ": " & childNode.Value
      Else
        childTreeNode = treeNode.Nodes.Add(childNode.Name)
        If childNode.HasChildNodes Then
          AddNode(childNode, childTreeNode)
        ElseIf childNode.Name = "StorageDevices" Then
          Dim storages() As String = childNode.InnerText.Split(New String()
{Environment.NewLine}, StringSplitOptions.None)
          For Each storage As String In storages
            childTreeNode.Nodes.Add(storage)
          Next
        Else
          childTreeNode.Text = childNode.Name & ": " & childNode.InnerText
        End If
      End If
    Next
  End Sub
  Private Sub ExportScanToolStripMenuItem_Click(sender As Object, e As EventArgs) Handles
ExportScanToolStripMenuItem.Click
    Try
      Dim saveDialog As New SaveFileDialog()
      saveDialog.Filter = "XML Files (*.xml)|*.xml"  ' Change filter to XML files
      saveDialog.Title = "Save XML File"

      If saveDialog.ShowDialog() = DialogResult.OK Then
        Dim xmlPath As String = saveDialog.FileName

        ' Check if the file already exists
        If System.IO.File.Exists(xmlPath) Then
          Dim overwriteResult As DialogResult = MessageBox.Show("The file already exists. Do
you want to overwrite it?", "File Exists", MessageBoxButtons.YesNo, MessageBoxIcon.Warning)
          If overwriteResult = DialogResult.No Then
            Return ' Exit if the user chooses not to overwrite
          End If
        End If

        Dim xmlContent As String = GetXmlStringFromTreeView(treeViewXML)

        If Not String.IsNullOrEmpty(xmlContent) Then
          ' Save XML content as XML file with line breaks for readability
          System.IO.File.WriteAllText(xmlPath, FormatXml(xmlContent),
System.Text.Encoding.UTF8)
```

```
            MessageBox.Show("XML file saved successfully.", "Success", MessageBoxButtons.OK,
MessageBoxIcon.Information)
          Else
            MessageBox.Show("XML content is empty or invalid.", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error)
          End If
        Else
          MessageBox.Show("No file selected.", "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error)
        End If
      Catch ex As Exception
        MessageBox.Show("An error occurred while saving the XML file: " & ex.Message, "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error)
      End Try
    End Sub
    Private Function FormatXml(xmlContent As String) As String
      Dim doc As New XmlDocument()
      doc.LoadXml(xmlContent)
      Using stringWriter As New StringWriter()
        Using xmlWriter As XmlWriter = XmlWriter.Create(stringWriter, New XmlWriterSettings()
With {.Indent = True, .NewLineOnAttributes = False})
          doc.Save(xmlWriter)
          Return stringWriter.ToString()
        End Using
      End Using
    End Function

    Private Function GetXmlStringFromTreeView(treeView As TreeView) As String
      Dim xmlDoc As New XmlDocument()
      Dim xmlDeclaration As XmlDeclaration = xmlDoc.CreateXmlDeclaration("1.0", "UTF-8",
Nothing)
      xmlDoc.AppendChild(xmlDeclaration)

      Dim rootNode As XmlNode = xmlDoc.CreateElement("SystemInformation")
      xmlDoc.AppendChild(rootNode)

      ' Add treeViewXML content to XML
      For Each treeNode As TreeNode In treeView.Nodes
        AddXmlNodeFromTreeView(xmlDoc, rootNode, treeNode)
      Next

      Return xmlDoc.OuterXml
    End Function
    Private Sub AddXmlNodeFromTreeView(xmlDoc As XmlDocument, parentNode As XmlNode,
treeNode As TreeNode)
      Dim newNode As XmlNode =
xmlDoc.CreateElement(SanitizeXmlName(treeNode.Text.Split(":")(0).Trim()))

      If treeNode.Nodes.Count > 0 Then
        For Each childNode As TreeNode In treeNode.Nodes
          AddXmlNodeFromTreeView(xmlDoc, newNode, childNode)
        Next
      Else
        newNode.InnerText = treeNode.Text.Split(":")(1).Trim()
```

```
        End If

        parentNode.AppendChild(newNode)
    End Sub
    Private Function SanitizeXmlName(name As String) As String
        ' Remove invalid characters from the name
        Dim sanitized As String = System.Text.RegularExpressions.Regex.Replace(name, "[^a-zA-Z0-
9_.-]", "_")

        ' Ensure the name starts with a letter or underscore
        If Not String.IsNullOrEmpty(sanitized) AndAlso Not Char.IsLetter(sanitized(0)) And Not
sanitized.StartsWith("_") Then
            sanitized = "_" & sanitized
        End If

        Return sanitized
    End Function

    Private Sub AddXmlElement(xmlDoc As XmlDocument, parentNode As XmlNode,
elementName As String, elementValue As String)
        Dim element As XmlElement = xmlDoc.CreateElement(elementName)
        element.InnerText = elementValue ' Ensure that this captures the RAM value as a string
        parentNode.AppendChild(element)
    End Sub

    Private Function GetOperatingSystemInfo() As String
        Dim osInfo As String = String.Empty
        Using searcher As New ManagementObjectSearcher("SELECT * FROM
Win32_OperatingSystem")
            For Each obj As ManagementObject In searcher.Get()
                osInfo = obj("Caption") & " " & obj("Version")
            Next
        End Using
        Return osInfo
    End Function

    Private Function GetOperatingSystemInstallDate() As String
        Dim installDate As String = String.Empty
        Using searcher As New ManagementObjectSearcher("SELECT * FROM
Win32_OperatingSystem")
            For Each obj As ManagementObject In searcher.Get()
                Dim dateStr As String = obj("InstallDate").ToString()
                installDate = ManagementDateTimeConverter.ToDateTime(dateStr).ToString("yyyy-
MM-dd")
            Next
        End Using
        Return installDate
    End Function

    Private Function GetCpuInfo() As String
        Dim cpuInfo As String = String.Empty
        Using searcher As New ManagementObjectSearcher("SELECT * FROM Win32_Processor")
            For Each obj As ManagementObject In searcher.Get()
                cpuInfo = obj("Name").ToString()
```

```
        Next
      End Using
      Return cpuInfo
    End Function

    Private Function GetRamInfo() As String
      Dim totalRam As Long = 0
      Using searcher As New ManagementObjectSearcher("SELECT * FROM
Win32_PhysicalMemory")
        For Each obj As ManagementObject In searcher.Get()
          totalRam += CLng(obj("Capacity"))
        Next
      End Using

      Dim totalRamInGB As Double = Math.Round(totalRam / 1024 / 1024 / 1024, 2)
      Return totalRamInGB.ToString() ' Return only the numeric value in GB
    End Function
    Private Function GetStorageInfo() As String
      Dim storageInfo As String = String.Empty
      Using searcher As New ManagementObjectSearcher("SELECT * FROM Win32_DiskDrive")
        For Each obj As ManagementObject In searcher.Get()
          Dim model As String = obj("Model").ToString()
          Dim size As Long = CLng(obj("Size"))
          Dim sizeInGB As Double = Math.Round(size / 1024 / 1024 / 1024, 2)
          storageInfo &= model & " (" & sizeInGB.ToString() & " GB)" & Environment.NewLine
        Next
      End Using
      Return storageInfo.Trim()

    End Function

    Private Function GetGpuInfo() As String
      Dim gpuInfo As String = String.Empty
      Using searcher As New ManagementObjectSearcher("SELECT * FROM
Win32_VideoController")
        For Each obj As ManagementObject In searcher.Get()
          Dim gpuName As String = obj("Name").ToString()
          Dim gpuMemory As Long = CLng(obj("AdapterRAM"))
          Dim gpuMemoryInMB As Double = Math.Round(gpuMemory / 1024 / 1024, 2)
          gpuInfo &= gpuName & " (" & gpuMemoryInMB.ToString() & " MB)" &
Environment.NewLine
        Next
      End Using
      Return gpuInfo.Trim()
    End Function
    Private Sub RunSystemPropertiesProtection()
      Try
        Dim system32Path As String =
Environment.GetFolderPath(Environment.SpecialFolder.System)
        Dim systemPropertiesProtectionPath As String = Path.Combine(system32Path,
"SystemPropertiesProtection.exe")

        If File.Exists(systemPropertiesProtectionPath) Then
          Process.Start(systemPropertiesProtectionPath)
```

```
        Else
            MessageBox.Show("SystemPropertiesProtection.exe not found in System32 folder.")
        End If
    Catch ex As Exception
        MessageBox.Show("An error occurred: " & ex.Message)
    End Try
End Sub


Private Sub btnWindowsSecurity_Click(sender As Object, e As EventArgs) Handles
btnWindowsSecurity.Click
    Process.Start("cmd.exe", "/c start windowsdefender:")
End Sub

Private Sub btnWindowsUpdate_Click(sender As Object, e As EventArgs) Handles
btnWindowsUpdate.Click
    Process.Start("cmd.exe", "/c start ms-settings:windowsupdate")
End Sub

Private Sub btnSystemRestore_Click(sender As Object, e As EventArgs) Handles
btnSystemRestore.Click
    RunSystemPropertiesProtection()
End Sub

Private Sub btnTaskManager_Click(sender As Object, e As EventArgs) Handles
btnTaskManager.Click
    Process.Start("cmd.exe", "/c start taskmgr")
End Sub

Private Sub btnDiskDefragmenter_Click(sender As Object, e As EventArgs) Handles
btnDiskDefragmenter.Click
    Process.Start("cmd.exe", "/c start dfrgui")
End Sub
Private Sub btnOpenAddOrRemovePrograms_Click(sender As Object, e As EventArgs) Handles
btnOpenAddOrRemovePrograms.Click
    RunAppwizCpl()
End Sub
Private Sub RunAppwizCpl()
    Try
        ' Use the Shell function to run appwiz.cpl
        Shell("appwiz.cpl", AppWinStyle.NormalFocus)
    Catch ex As Exception
        MessageBox.Show("An error occurred: " & ex.Message)
    End Try
End Sub

Private Sub btnPing_Click(sender As Object, e As EventArgs) Handles BtnPing.Click
    Dim address As String = txtAddress.Text.Trim()

    ' If no address is entered, ping Google.com
    If String.IsNullOrEmpty(address) Then
        address = "google.com"
    End If
```

```
        If Not String.IsNullOrEmpty(address) Then
            LstPingResults.Items.Clear() ' Clear the ListBox before performing the ping

            Dim processInfo As New ProcessStartInfo("ping", "-n 4 " & address) ' Ping the address 4
times
            processInfo.RedirectStandardOutput = True
            processInfo.UseShellExecute = False
            processInfo.CreateNoWindow = True

            Dim process As Process = Process.Start(processInfo)
            process.WaitForExit()

            Dim output As String = process.StandardOutput.ReadToEnd()

            ' Split the output by line and add it to the list box
            For Each line As String In output.Split({Environment.NewLine},
StringSplitOptions.RemoveEmptyEntries)
                LstPingResults.Items.Add(line)
            Next
        Else
            MessageBox.Show("Please enter an address to ping.")
        End If
    End Sub

    Private Sub Button1_Click(sender As Object, e As EventArgs)
        Process.Start("DISMCheckhealth.bat")
    End Sub

    Private Sub Label6_Click(sender As Object, e As EventArgs) Handles Label6.Click

    End Sub
End Class
```

End of Source Code for main window functionalities.

```
@echo off
:menu
cls
echo Choose an option:
echo +---------------------------------------------------------------------------------------------------+
echo 1. DISM Scan health
echo     Scans the Windows image for corruption or integrity issues
echo     Useful when troubleshooting system problems or verifying the health of the Windows
installation.
echo +---------------------------------------------------------------------------------------------------+
echo 2. DISM Restore health
echo     Restores the Windows image by repairing any corrupted files found during the scan.
echo     Helps fix issues related to missing or damaged system files.
echo +---------------------------------------------------------------------------------------------------+
echo 3. DISM Check health
echo     Checks the health of the Windows image without attempting repairs.
echo     Provides a quick assessment of the system's health.
echo +---------------------------------------------------------------------------------------------------+
echo 4. SFC Scan now
```

```
echo     Scans and repairs protected system files using the System File Checker (SFC) utility.
echo     Useful when encountering issues related to system file corruption.
echo +---------------------------------------------------------------------------------------------------+
echo 5. WMIC Get hard disks status
echo     Retrieves information about the status of hard disk drives.
echo     Helps monitor disk health, check for errors, and identify failing drives.
echo +---------------------------------------------------------------------------------------------------+
echo 6. Exit

set /p choice=Enter the number of your choice:

if "%choice%"=="1" (
   start cmd /k dism /Online /Cleanup-Image /ScanHealth
   goto menu
) else if "%choice%"=="2" (
   start cmd /k dism /Online /Cleanup-Image /RestoreHealth
   goto menu
) else if "%choice%"=="3" (
   start cmd /k dism /Online /Cleanup-Image /CheckHealth
   goto menu
) else if "%choice%"=="4" (
   start cmd /k sfc /scannow
   goto menu
) else if "%choice%"=="5" (
   start cmd /k wmic diskdrive get status
   goto menu
) else if "%choice%"=="6" (
   exit
) else (
   echo Invalid choice. Please try again.
   pause
   goto menu
)
```

End of source code for external command prompt tool